Fischer, J.; Redlich, J.; Scheuermann, B.; Schiller, J.; Günes, M.; Nagel, K.; Wagner, P.; Scheidgen, M.; Zubow, A.; Eveslage, I.; Sombrutzki, R.; Juraschek, F.

From Earthquake Detection to Traffic Surveillance – About Information and Communication Infrastructures for Smart Cities

The final authenticated version is available online at https://doi.org/10.1007/978-3-642-36757-1_8.

Fischer, J.; Redlich, J.; Scheuermann, B.; Schiller, J.; Günes, M.; Nagel, K.; Wagner, P.; Scheidgen, M.; Zubow, A.; Eveslage, I.; Sombrutzki, R.; Juraschek, F. (2013). From Earthquake Detection to Traffic Surveillance – About Information and Communication Infrastructures for Smart Cities. Lecture Notes in Computer Science, 121–141. https://doi.org/10.1007/978-3-642-36757-1_8

From Earthquake Detection to Traffic Surveillance – About Information and Communication Infrastructures for Smart Cities

Joachim Fischer[1] Jens-Peter Redlich[1], Björn Scheuermann[1], Jochen Schiller[2], Mesut Günes[2], Kai Nagel[3], Peter Wagner[4]
Markus Scheidgen[1], Anatolij Zubow[1], Ingmar Eveslage[1], Robert Sombrutzki[1], and Felix Juraschek[2]

[1] Humboldt Universität zu Berlin
{fischer,redlich,scheuermann,scheidge,zubow,sombrutz,eveslage}
@informatik.hu-berlin.de
[2] Freie Universität Berlin
{jochen.schiller,mesut.guenes,felix.juraschek}@fu-berlin.de
[3] Technische Universität Berlin
kai.nagel@tu-berlin.de
[4] Deutsches Zentrum für Luft und Raumfahrt
peter.wagner@dlr.de

Abstract. Smart cities use networks of sensors, actuators, and centralized computing clusters to observe physical reality, derive information, and thereby influence citizens and authorities. Smart city applications therefore require three components to work: wireless sensor networks, geo-information systems, and frameworks for distributed analysis of sensor and geo-data. In this paper, we provide an overview on a set of concrete technologies for such information and communication infrastructures for smart cities. These technologies include a combination of WiFi- and PAN-based sensor networks, City GML data, a model-driven approach to collect and manage data, as well as distributed data analysis based on domain specific languages. We show how we use these technologies to research two typical smart city applications: earthquake early warning and traffic surveillance.

1 Introduction

Smart cities use networks of sensors, actuators, and centralized computing clusters to observe physical reality, derive information, and thereby influence citizens and authorities. A series of smart city applications was discussed the last years: *SmartGrids* [33,21] use *SmartMeter* and dynamic energy prices to help consumers use electricity for efficiently. Wireless sensor networks in *earthquake early warning systems* [8] detect earthquakes quickly and automatically shutdown cities energy and traffic infrastructure. Smart parking space control systems use sensors to direct drivers to free parking spots [13]. Many cities already maintain wireless sensor networks to research such applications. Two

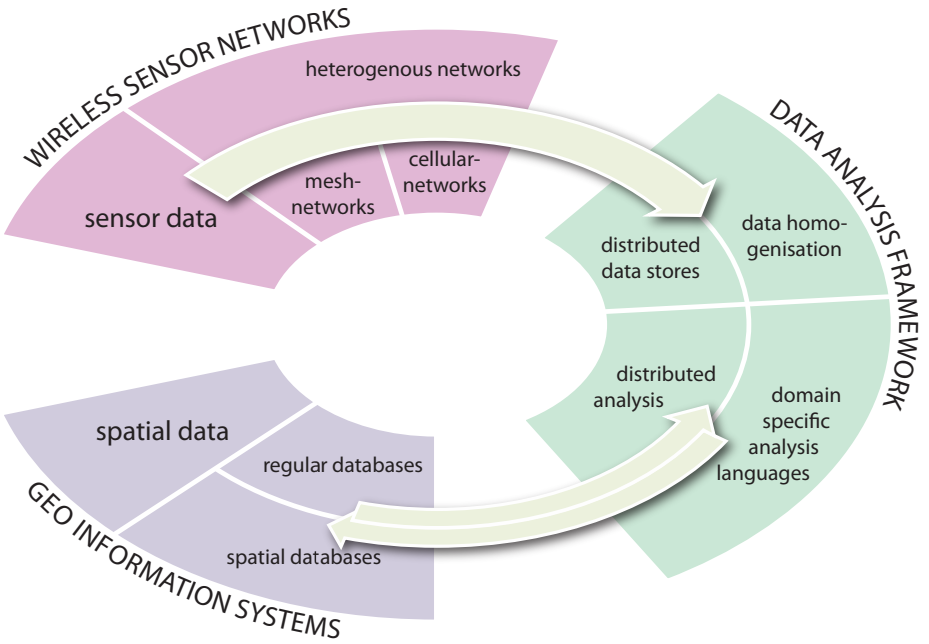


Fig. 1: Smart City technology overview

examples are *CitySense* in Bosten, USA [19] and *SmartSantander* in Santander, Spain [13,5].

All these applications and services require three fundamental technology components. First, *wireless sensor networks* (WSN) that aquire data about the physical reality and privide the means to transport data. Secondly, *geo information systems* (GIS) provide context data necessary to interprete sensor data. Thirdly, we need *data analysis frameworks* that can process large amounts of heterogeneous data in complex chains of individual computation steps. These components are depicted in Fig. 1.

In this paper, we describe the combined efforts of researchers at Berlin's three major university to develop an infrastructure that provides these three components and therefore allows researchers to build smart city application and services. The paper is organized as follows. The next three sections present the used sensor networks, the used data standards and geo information systems, and our approach on analysing the vast amount of expected data by modern software engineering means. The next two sections present our efforts in research two smart city applications: earchquake early warning and traffic surveillance. We end the paper with a discussion of possible future developments and implications.

2 Wireless Sensor Networks

Wireless sensor networks (WSNs) [7] are battery powered wireless multi-hop networks, where each node is equipped with multiple sensors. WSNs allow to sense the environment without any existing infrastructure. Typical WSNs use low powered and energy-efficient hardware with short-range radio communication (e.g. tmote sky) typically based on Wireless Personal Area Networks (WPAN), e.g. IEEE 802.15.4. As main characteristic these WSNs use short duty cycles and long periods of inactivity to preserve batteries. Thus, WSNs are tailored for measurements at low sensor sample rates or over short periods of time. Due to high energy consumption of radio communication, WSNs typically record data locally and communicate only aggregated data of small size (Fig. 2, left).

Typically WSNs are used to measure a single physical variable. Applications include measurement of temperature, sensing the presence or absence of objects, monitoring the structural health of buildings via vibrations and natural frequencies (Structural Health Monitoring (SHM) [18]), or sensing the acoustic stress by measuring noise levels [2].

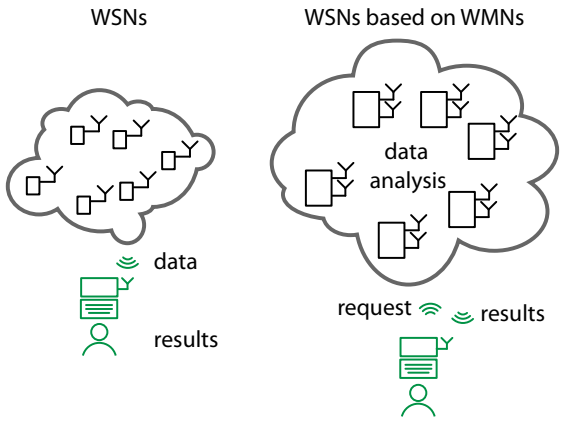


Fig. 2: The HWL-Testbed enables the development and analysis of applications for HP-WSN.

WNS applications are limited by computation and networking capabilities of WSNs. Applications such as SHM with higher sample rates at multiple channels, recording of audio (e.g. speech in contrast to just noise levels), measuring patterns such as bitmaps created by video cameras [1], or measuring multiple properties at once (e.g. detecting correlation between temperature and natural frequencies in concrete) require a different kind of WSN. Such applications produce different types of data in large amounts in a short period of time.

WSNs based on Wireless Local Area Network (W-LAN), e.g. IEEE 802.11 with a larger physical size, higher battery weight, or even cable based power

supply are a new class of sensor networks based on wireless mesh networks (WMNs). Compared to managed wired networks these ad-hoc WMNs still allow fast and cost effective way to install a communication and sensing infrastructure in a previously unknown and changing environment. Beyond sensing, this WMNs provide enough capabilities to run data analysis within the network (Fig. 2, right).

We use combinated WSN/WMN testbeds that comprise both kinds of sensor nodes. Our two test-beds at the Humboldt and Freie University in Berlin form a interconnected network configuration. A network configuration [10] is an architecture that describes the way how different kinds of networks can be connected and integrated to support particular applications and services for Smart Cities. In the following, we describe existing network configurations for WMNs and WSNs and how those configurations can be recombined into a heterogeneous network configuration for smart cities.

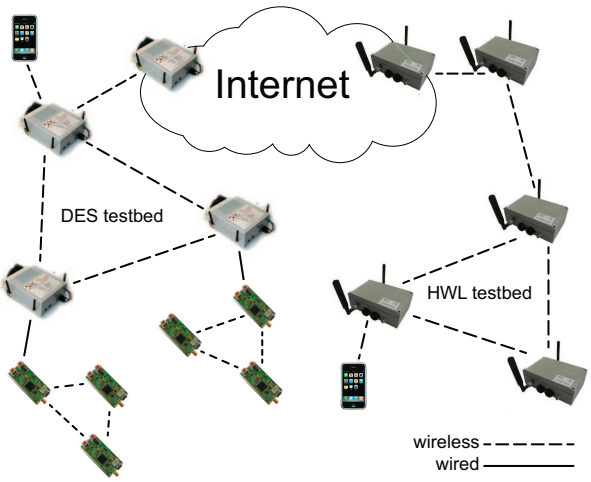
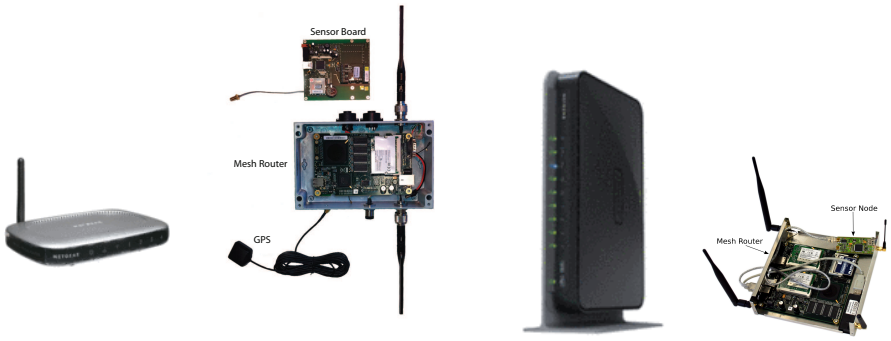


Fig. 3: Integrated DES- and HWL-Testbeds

The basic building blocks of smart city network configurations are *WMNs* and *WSNs*. Both are self-organized mesh networks that provide autonomous configuration, self-healing and basic deployment capabilities. WMNs are traditionally studied as stationary or mobile (mobile ad-hoc networks – MANET) access networks for mobile wireless clients. WMNs are typically based on 802.11x hardware; WMN nodes have larger radio range, higher computational capabilities, run more complex software and require more power than the nodes of a WSN. WSNs on the other hand are only used to monitor the environment with sensors. They are smaller, only wake for short sensing cycles, and only communicate to provide sensor data to clients. Nodes of traditional WMNs can also



(a) HWL v1 (b) HWL v2 (c) HWL v3 (d) DES

Fig. 4: Three different types of HWL nodes are used: (i) HWL v1 which is an indoor node with a single 802.11b/g radio device, (ii) HWL v2 which is an outdoor node with two 802.11a/b/g radio devices and (iii) HWL v3 is an indoor node with two 802.11a/b/g/n (MIMO) devices, and the DES node (iv) comprising the multi-radio mesh router and the MSB-A2 sensor node.

be equipped with sensors to create *high performance WSNs* [28] in contrast to typical WSNs.

Fig. Figure 3 shows the integrated DES- and HWL-Testbeds as a combination of WMNs and WSNs with WMNs acting as integrator between WSNs and access network for clients; the Internet provides an interconnection between the two networks. In the following, we first describe the HWL-Testbed at HU Berlin and the DES-Testbed at FU Berlin and their features.

2.1 HWL-Testbed

The Humboldt Wireless Lab (HWL) is a large-scale wireless mesh network at the campus of the Humboldt University, Germany. It consists of about 125 mesh nodes based on 802.11a/b/g as well as the new 802.11n standard which are deployed indoor as well as outdoor. The indoor nodes, which are placed in several buildings, form a fully connected wireless network, which can be combined with the outdoor network to improve the connectivity between the buildings. The aim of HWL is to evaluate large-scale mesh networks, since small- and medium-scale mesh networks are already well understood. The upcoming IEEE 802.11n standard promises to significantly increase coverage, reliability, and throughput which comes from the advanced antenna technology based on Multiple Input Multiple Output (MIMO) techniques.

All indoor nodes are connected via a wired VLAN backbone to a central testbed server, which provides services like TFTP, DHCP, DNS and NFS. In

contrast the outdoor nodes are connected by a wireless mesh network backbone and a gateway with the testbed server. Therefore the second wireless interface is used and cannot be used for experiments. All experiments are centrally controlled from the testbed server where also the data collected in the experiments is stored, which simplifies the analysis considerably.

The HWL mesh software (adressing, routing, physical layer rate control, etc.) is implemented using the Click router API [16]. A Click router is built by sticking together several packet processing modules, called elements, forming a directed flow graph. Each element is responsible for a specific task such as packet classification, scheduling, or interfacing with networking devices. A detailed technical description of the used hardware, software and testbed architecture is available [36] and [26,27].

2.2 DES-Testbed

The *Distributed Embedded Systems Testbed* (DES-Testbed) is a hybrid wireless network located on the campus of Freie Universität Berlin. Currently, 128 DES-Nodes are available with future upgrade plans to a total of 150. It is hybrid in a way, that all DES-Nodes consist of a wireless mesh router equipped with multiple IEEE 802.11a/b/g radios and a MSB-A2 sensor node [3] as shown in 4d. Thus, a WMN based on IEEE 802.11 technology, called DES-Mesh, and a WSN, called the DES-WSN, are operated in parallel. In this manner it is possible to constitute all the possible network configurations for *wireless multi-hop networks* (WMHNs).

The DES-Nodes are scattered in an irregular topology across several buildings on the campus. Most of the nodes are deployed inside the offices, while some outdoor nodes are added to improve the connectivity and increase the approximation to real world scenarios. A testbed server DES-Portal functions as the central control instance in the DES-Testbed. It is connected to all DES-Nodes via an Ethernet backbone. A detailed technical description of the used hardware and testbed architecture is available in our technical reports [11] and [12].

For the purpose of comparison and easy implementation of routing protocols we developed the *Distributed Embedded Systems - Simple and Extensible Routing-Framework for Testbeds* (DES-SERT) [4]. The daemon forwards Ethernet frames or raw IP datagrams in an underlay (layer 2.5 routing, like in MPLS) so that the routing is transparent to the upper layer protocols. As mobility is an important aspect for the research on WMNs and WSNs, DES-SERT has also been ported to the Android smartphone platform. An Android-based smartphone mounted on a Lego NXT robot serves as a mobile client to the testbed.

3 Geo Information Systems

To interpret sensor data correctly the context of where and when data was acquired is important. Thermometer readings taken in the shade have different

meaning than readings taken in direct sunlight. Therefore, we need a system that can provide context data as further input for sensor data analysis.

A geo(graphic) information system (GIS) is a system designed to capture, store, manipulate, analyze, manage, and present all types of geographical data. GIS data represents real objects (such as roads, land use, elevation, trees, waterways, etc.) with digital data. Real objects can be divided into two abstractions: discrete objects (e.g. a house) and continuous fields (such as rainfall amount or elevations). There are two methods used to store data in a GIS for both kinds of abstractions: vectors and raster images.

In our work and in this paper, we concentrate on vector data. There is a series of OGC (Open Geo-Spatial Consortium) standards to represent geographic vector data for different applications. Two of them are the Geographic Markup Language (GML) [20] and City GML [20]. Both standards are based on XML and XML-Schema. While GML defines a basic set of XML-types to describe arbitrary geographic objects by means of their location, shape, and composition, City GML is specialized to model cities. City GML defines specific types to define buildings, structure, furniture, and their parts. One of City GML's key features is that it is not limited to modeling the 3D properties of objects, but that it also allows its users to define semantic extensions. These semantic extensions can be used to add all kinds of related information to objects such as materials, usage, legal, or census data.

Therefore, City GML can be used to answer questions like: from which windows of which rooms in which buildings can I have free views on certain places, streets or monuments? To what floor the building were affected by a flood in each case? Which buildings of a special district have roofs, which are oriented to the south with a special angle of inclination. There are three approaches to technically serve City GML data. Relational databases (with geo-spatial extensions) with proprietary interfaces [30], managed sets of XML-files [17], or as fragmented EMF-models [24].

Since we already manage sensor data based on fragmented EMF-models and the goal is to relate City GML data with this sensor data, it makes sense to use the last approach. City GML's XML-schemata can be used to automatically derive corresponding EMF-meta-models. Therefore, City GML data can be represented as EMF-models, it can be managed with EMF-Fragments (refer to the next section), and it can be created, modified, accessed, and deleted with EMF's generated and reflection interfaces. The combination of EMF, EMF-fragments, City GML schemata therefore forms a GIS for City GML data that we can use within the same data analysis framework that we use to store, manage, and analyze sensor data. Refer to the next section for more details.

4 Data Analysis Frameworks

There are three goals of a data analysis framework. First, heterogeneous sets of data from different sources (e.g. WSNs and GISs) can be persisted and organized automatically on a cluster. Secondly, complex computations can be de-

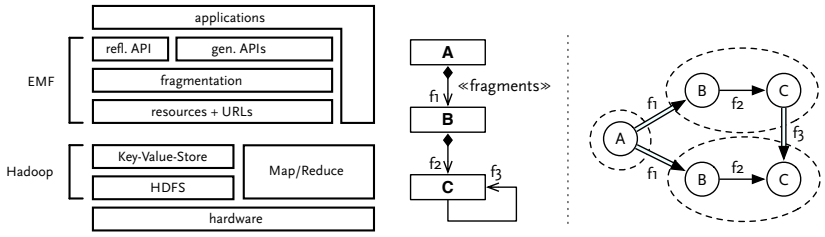
scribed with data-type specific concepts and independent from the details of their distributed execution. Thirdly, the logical connections between input and output of each computation step are recorded to allow for later interpretation of the potentially vast amounts of individual results. To achieve these goals, we apply meta-modeling and model transformation based on EMF to distributed data processing based on Apache's HBase and Hadoop. Fig. 5a depicts the architecture overview. In the following subsections, we describe three necessary steps: collecting data from WSNs, organizing data in a distributed data store, and finally running computations on this data.

4.1 Collecting Sensor Data

Both testbeds provide means to design and control experiments, to collect corresponding data and organize it in a central data store.

DES-Testbed Management System: The *DES-Testbed Management System* (*DES-TBMS*) comprises all steps of an experiment, namely the definition, automatized execution, and evaluation of experiments. *DES-Cript* is a domain specific language (DSL) based on XML, which defines and describes network experiments in a holistic way. *DES-Exp* provides an experiment manager which is responsible for the scheduling and execution of experiments. *DES-Web* provides a web interface to *DES-Exp*, which allows to create, modify, and schedule experiments using *DES-Cript*. The network monitoring tool *DES-Mon* is based on SNMP and retrieves the network state from the *DES-Nodes*. *DES-Mon* collects data from the wireless interfaces, the kernel routing table, ETX neighborhood information, and data from the sensor nodes. *DES-Vis* is a 3D-visualization software based on the JavaView framework. The evaluation tool *DES-Eval* enables the post-processing of the experiment results supporting work flows for an automatic evaluation process.

HWL-ClickWatch: ClickWatch [23] is an experiment control and analysis framework. ClickWatch connects to the Click runtime installed on all HWL-Testbed nodes and constantly collects data provided by all software components on the node (e.g. sensor, network protocol, and system components). This data comprises sensor data, network and system statistics, and configuration parameters. ClickWatch allows to visualize and change the nodes internal state and configuration at runtime. Furthermore, ClickWatch transforms incoming heterogeneous data into a homogeneous strongly typed representation. ClickWatch stores data in an HBase database and allows to access this data through statically typed APIs. This allows to write safe, reusable analysis scripts. ClickWatch represents all data within the Eclipse Modeling Framework (EMF). EMF based parser tools allows to transform the log-file style data provided by the DES-testbed or other 3rd-party networks into the same structured ClickWatch representation. This allows an integrated analysis of data provided by both the Smart Berlin networks: DES and HWL.



(a) Architecture (b) Fragmentation

Fig. 5: Overview of our architecture and the idea of model fragmentation.

4.2 Homogenisation and Distributed Organisation of Data

Our sensor networks and the used GISs provide complex sets of interconnected data based on a large amount of different types and based on different data modeling methodologies (EMF, XML, log-files, CSV-files, etc.). Dealing with complex, interconnected, well typed data-structures is the core trait of model driven software engineering. Meta-models allow to define fine grained object oriented types and references; constraints can elaborate meta-models, and semantics can be assigned to data structures. There is a large zoo of model transformation languages, programming frameworks, and other techniques. Software modeling (especially model driven architecture) provides the tools to integrate different typed structures (typically called languages). Furthermore, software modeling (especially EMF-based) integrates well with other core technologies like XML, ontologies, or even databases (e.g. via ORMs like CDO). But, software models are usually small enough to fit into main memory and scaleablity is less of an issue. Therefore, there are two challenges. First, we need to determine how meta-modeling can be used to integrate the different data sets. Secondly, we need to extend existing meta-modeling frameworks to handle large amounts of data.

Meta-Modeling as Integrator for Different Kinds of Data: If we look at our three data sources, we have EMF-based data, text/log-file based data, and XML-data. EMF- and XML-based are covered through modern meta-modeling frameworks like EMF. EMF-data explains itself, and XML-data can be intgrated since EMF use XML as native persistence format. Text- and log-file-based needs some efforts to describe its formal structure. We can use text-to-model transformation techniques to create parsers that extract data from text-based files and create an EMF-based representation of this data.

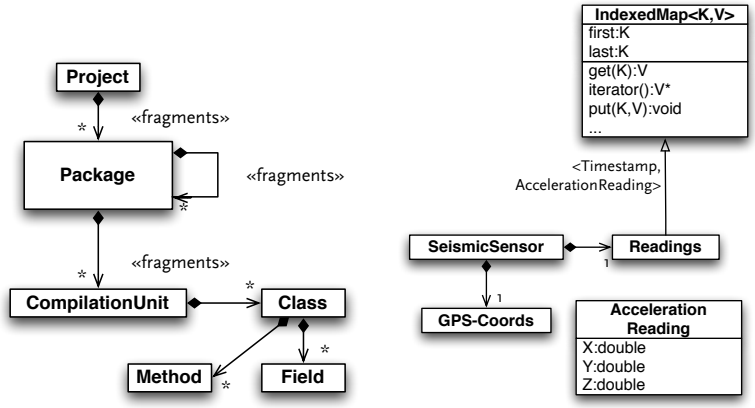
Distributed Storage of Large Meta-Model-based Data Sets: We build a model persistence framework for EMF [31] called *EMF-fragments* [22,25]. EMF-Fragments is different from frameworks based on object relational mappings (ORM) like Connected Data Objects (CDO) [32]. While ORM mappings map single objects, attributes, and references to database entries, EMF fragments map larger chunks

of a model (fragments) to URIs. This allows to store models on a wide range of distributed data-stores including distributed file-systems and key-value stores (e.g. Hadoop's HBase).

EMF-Fragments uses and extends the regular EMF resource API. Clients designate references that shall fragment the model in the meta-model. Fig. 5b exemplifies fragmentation on meta and model level. EMF-Fragments then automatically and transparently creates and manages resources and their content. This allows to control fragmentation without the need to trigger it programatically. Fragments/resource are continuously managed in the background, i.e. resources are loaded and also unloaded as necessary. Each fragment is backed by an EMF resource and identified by its URI. Resources and URIs are canonically mapped to keys and values in a key-value store (e.g. Hadoop's HBase). From the client perspective one just uses the regular reflective (refl.) or generated (gen.) EMF-APIs.

There are two general ways to describe fragmentation in the meta-model. The first one is to mark containment reference in the Ecore meta-model with annotations. This tells EMF-fragments to create a new resource for each value in those references. This works well when the number of anticipated values per feature is relatively low. This is usually the case in software models. In a large Java code base for example, we have a large number compilation units (i.e. Java class files), but a single package only contains a small set of sub-packages and compilation units (example in Fig. 6a).

With the described method the container has to keep references to all its contents. If we have millions of values the references alone require too large resources even though the values are stored in different resources. Therefore, we



(a) Reference fragmentation (b) Index fragmentation

Fig. 6: Examples for the different methods for describing model fragmentation.

also need a second approach: clients can use predefined index classes to define relationships with large value sets. This happens for example, if we want to store sensor readings in EMF: each sensor might have million of readings depending on the period of time and sample rate (example in Fig. 6b).

An index is simply a sorted list of key-value pairs. Indices are mapped to the underlying key-value store. Therefore, the used key-value store has to be sorted (e.g. like Hadoop's HBase). The index managing class stores the first and the last key of the index. Elements can be accessed directly using keys or they can be iterated.

4.3 Distributed Computation of Data

The Hadoop web site describes it's Map/Reduce capabilities like this: "Map-Reduce is a software framework for easily writing applications which process vast amounts of data (multi-terabyte data-sets) in-parallel on large clusters (thousands of nodes) of commodity hardware in a reliable, fault-tolerant manner." Map/Reduce has proven itself as a successful programming model. But, Map/Reduce has two known disadvantages. First, Map/Reduce is only suited for so called *embarrassingly parallel problems*. Engineers struggle to implement algorithms for problems that do not canonically fall into independent parts. Secondly, Map/Reduce deliberately ignores the structure of the data that it is used to analyze. This issue is delegated to users. The consequences are many fold; examples are hardly reusable algorithms based on proprietary data-structures or slow, defective, and proprietary parsers.

We use EMF to help with the second problem: the EMF-Fragments framework introduced in the prior section stores data as fragmented EMF models in a key-value-store, and Map/Reduce implementations are designed to work with these stores. We are using Hadoop as an implementation of Map/Reduce, and we use Hadoop's HBase key-value-store as our datastore. In the Hadoop/HBase framework, clients have to extend abstract Map and Reduce classes to define their own Map/Reduce function pairs. Abstractions for input and output of these functions are arbitrary text files (stored with Hadoop's distributed file system HDFS), or key-value entries (stored in HBase tables). We extend these Hadoop/HBase's abstractions to use EMF objects as input and output. Instead of raw values, clients are given the EMF contents of the corresponding resources, and instead of writing raw values, clients can create new EMF-objects.

Despite all this convenience, users still have to put a lot of thought into their problems. First, clients have to design their meta-models intelligently (i.e. create reasonably fragments). The goal is to create fragments that allow each map task to only work with a single fragment. This allows Hadoop to preserve locality and execute map tasks on nodes that already store their input data. Secondly of course, only a small set of problems maps to the Map/Reduce paradigm trivially.

5 Application I: Earthquake Early Warning

Disasters caused by natural phenomena are considered as one of the most threatening events of today's modern world. Even though most of them cannot be predicted, efforts can be made for mitigating human and economic losses. This can be achieved by means of early warning, which allows individuals exposed to hazard to take action to avoid or reduce their risk and prepare for effective response. In this context, the main challenge is to minimize the delay between the detection of an occurred event and the delivery of alarm messages in order to maximize the time available for preventing possible damages. Nevertheless, the development of reliable infrastructures for supporting early warning is not trivial because of the diversity of the natural phenomena and cost related issues.

Earthquake Early Warning (EEW), as a special case of early warning, is characterized by a very short delay between the actual earthquake event and its destructive impact. Current EEW Systems (EEWS) are composed of few expensive and highly sensitive sensor stations, installed outside of urban areas, and connected to a single data center. These centralized infrastructures have a number of problems related to single point of failure, insufficient node density, and an increased delay between event and warning propagation caused by the missing integration into the target area.

In contrast to existing EEWS, our approach, the Self-Organizing Seismic Early Warning Information Network (SOSEWIN) [9], is technically a decentralized, self-organizing wireless mesh network (WMN), equipped with seismological sensors, made up of low-cost components. With a relatively low price, a very dense network (hundreds or thousands of nodes) can be established directly in the threatened regions. A new question connected with a self-organized warning system concerns the potential end user of the warnings. An alarm could be more than the information of a centralized disaster management facility; it can be used for a direct information of the private owners and their neighborhoods, a decentralized control (power-downs, gas pressure reduction) of technical devices, plants and more.

SOSEWIN realizes EEW by means of a distributed application with hard real-time constraints, raising the early warning inside the network itself. We follow a generally approved model-driven development paradigm using standardized languages to generate code for the target platform (sensor nodes) and for different kinds of simulators. These simulators are combined with environment models in order to evaluate the early warning performance of the network. The environmental model consists of synthesized timed data series as imitations of the ground shaking for each chosen geographical sensor node position in dependence to the distance of the epicenter and the magnitude of that imaginary event.

Earthquakes produce different types of seismic waves, which travel from the hypocenter in every direction. Their analysis is the foundation for different activities in disaster management (i.e. earthquake classification, early warning, and first response). There are four types of seismic waves divided into two groups:

- P-waves and S-waves (called body waves). They travel through the interior of the Earth. P-waves (primary waves) travel faster than S-waves (secondary waves) [5]. They are less destructive than the S-waves and surface waves that follow them.
- Rayleigh waves and Love waves (called surface waves). They remain below the Earth's surface and can be much larger in amplitude than body waves.

Even though it is not possible to predict an earthquake event, preparations can be made for the incoming disaster. This can be achieved by using the time delay between the arrival times of the P-wave and S-wave (Figure 7). This delay varies from a few seconds to some minutes depending on the distance between the epicenter of the earthquake and the critical area locations.

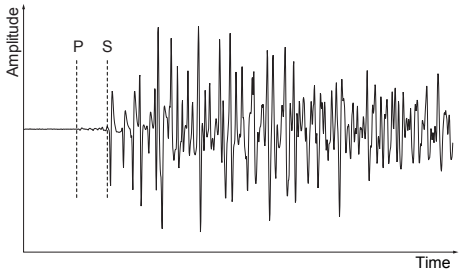


Fig. 7: Time delay between P-wave and S-wave.

Nowadays Earthquake Early Warning Systems (EEWS) are based on the detection of the harmless P-waves that precede the slower and destructive S-waves and surface waves. Therefore, the primary goal of an EEWS is to maximize the early warning time under a minimal number of false alarms (false positives and false negatives).

An important secondary goal is the fast generation of the so-called shake maps [34] for affected regions, which show the maximal ground shaking in a dense grid. The combination of such maps with information about building structures and population densities in the affected area is important for fast and proper disaster management.

Almost all current EEWS use a centralized approach (e.g. Taiwan: [35]; Japan: [14]; Istanbul: [6]; Bucharest: [15]). Each station delivers its measured data over a direct connection to a central data center. These EEWS often consist of only a few, but expensive stations (several thousands of Euro), resulting in a number of problems:

[5] Dependent upon the geology of the specific region and the hypocenter depth, P-waves travel at 5-8 km/s, and S-waves at 3-7 km/s.

- Malfunction: If one station breaks down, then the area it would normally observe can only be monitored from afar, resulting in time delays that could seriously compromise the network's early warning capacity.
- Density: This problem is related to the generation of precise information about an earthquake's intensity for city square cells, generally in size of 500 m. By comparison, EEWS usually have a station spacing of several kilometers.
- Cost: However, increasing the density of seismic stations is limited by their expense.
- Communication: The reliable transmission of all station information to central data center or civil protection headquarters is very important, especially following an earthquake, where usually centralized communication infrastructures may have collapsed.

Our approach addresses the problems identified above by deploying a much higher number of much cheaper stations (costing only a few hundreds of Euros). This approach is based on a wireless mesh network, where each node is equipped with the necessary components.

The reliability of such an EEWS is improved since the system can detect an earthquake even though single sensors may have been destroyed. This can be achieved because the sensor nodes act cooperatively in a self-organizing way.

The approach of equipping WMN nodes with sensors leads to a EEWS that can be deployed cheap and easily in threatened cities. The essential principle is the cooperative signal analysis done in the network and the availability of several services for self-organizing management, enabling the distinction between medium earthquakes and other events in urban environment like construction sites or trains. Centralized disaster management facilities as well as the possessors of our nodes can be directly informed by such a system. This creates a new culture of early warning where everyone can participate. Cooperative signal analysis and alarming can be used for other early warning use cases which require a sensing acquisition of environmental phenomena under real-time constraints.

Additionally to our HWL testbeds in Berlin, we deployed a test-bed of 20 nodes in Istanbul. The testbed in Istanbul with its difficult conditions and the missing direct access to the nodes led to the development of a collection of remote administration and experiment management tools. The HWL testbed with its reliable network topology allowed a repetitive analysis and performance evaluation of the alarming protocol for EEW. Still, the sensitivity for false-alarms in a noisy environment has to be studied.

While we have realized and shown that earthquake early warning is possible with such a system we are still lacking robustness to achieve the real-time constraints in changing network conditions. In this context, the improvement of transport and routing protocols, link metrics and topology optimization is still an open research issue.

6 Application II: Traffic Surveillance

6.1 Motivation

Traffic infrastructure isn't build in abundance and temporal overload due to regular and periodical spikes in traffic volume or due to extra ordinary causes (e.g. accidents) is normal. To understand this behavior, many traffic models differentiate between classes of vehicles. Different travel purposes are assigned to different classes and consequently different classes of vehicles show different behavioral patterns [29].

Stationary detection systems (e.g. induction loops, traffic-Eyes) and Floating Car Data (FCD) are today's preferred tools to collect the data that is fed into respective traffic models. For economic and data privacy reasons these techniques cannot be extended arbitrarily and not all techniques excel at vehicle classification. As a result coverage and quality of traffic data is unsatisfactory. To implement innovative traffic management and control methods, a precise knowledge of the current traffic situation and a reliable prediction of future traffic situations is required.

The use of commodity measurement instruments based on meshed sensor networks can be a viable new method, provided we can improve the quality of vehicle identification and classification while securing anonymity. Therefore, we currently apply our acceleration sensor based WSN to the detection of road fright traffic.

6.2 Experiments

In this section, we report on our first experimental results using HWL sensor nodes for traffic surveillance.

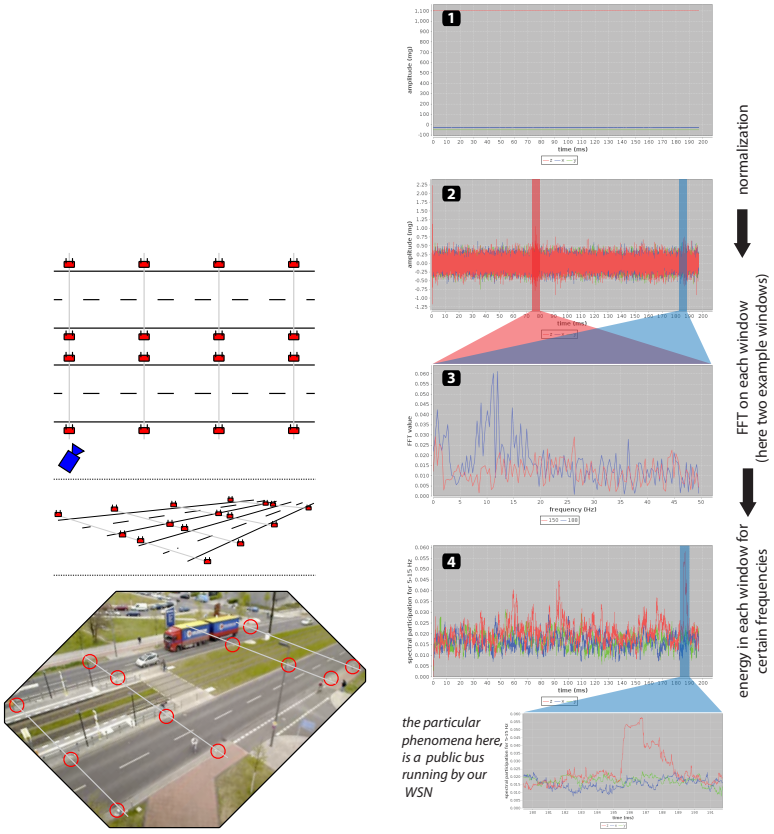
Methodology: The basic approach is to validate acceleration sensor data and the results of corresponding analysis algorithms against reference data to determine statistical quality criteria (false-positive rate, sensitivity, specificity, etc.). The input of each experiment is a specific road, a sensor network, and a set of analysis algorithms; the output is a set of quality criteria for the given algorithm.

We choose actual roads or empty test roads of different difficulty: varying number of lanes, varying traffic patters, controlled (test roads) or actual traffic. We deploy four sensor nodes at each side of the road (if applicable also on the median strip) at an equal distance of several vehicle lengths. Each sensor node is equipped with a 3-axis accelerometer and GPS (for time and position). Additionally a video camera is deployed. The camera has a global view on all sensors and the corresponding parts of the road.

Sensor data and video feed are recorded for a period of time. Video data is manually analyzed and a formal (computer-understandable) transcript of the traffic is produced. This transcript determines what vehicles (based on a prior classification) have passed which sensor at what time. All algorithms are applied to the recorded sensor data. The algorithms are designed to produce output of

similar structure to the video transcripts. Analysis output and transcripts are used to compute the statistical quality criteria. Fig. 8a shows our experiment setup at a four-lane road.

Algorithms: We developed and analyzed several algorithms with different complexity. There are algorithms that only compute the input of a single node, algorithms that use data from a neighborhood of nodes, or even all nodes. Algorithms can analyze in time and frequency domains. Typical operators used in our algorithms includes Fast Fourier Transformations (FFTs), binning, sliding windows, band filters, calculating statistical moments, etc. It is generally favorable to express these algorithms in a language that allows for mathematical expressions and libraries that supports the identified operations. Fig. 8b shows the different steps of an FFT-based algorithm.



(a) Setup (b) Analysis steps

Fig. 8: Experiments on traffic surveillance.

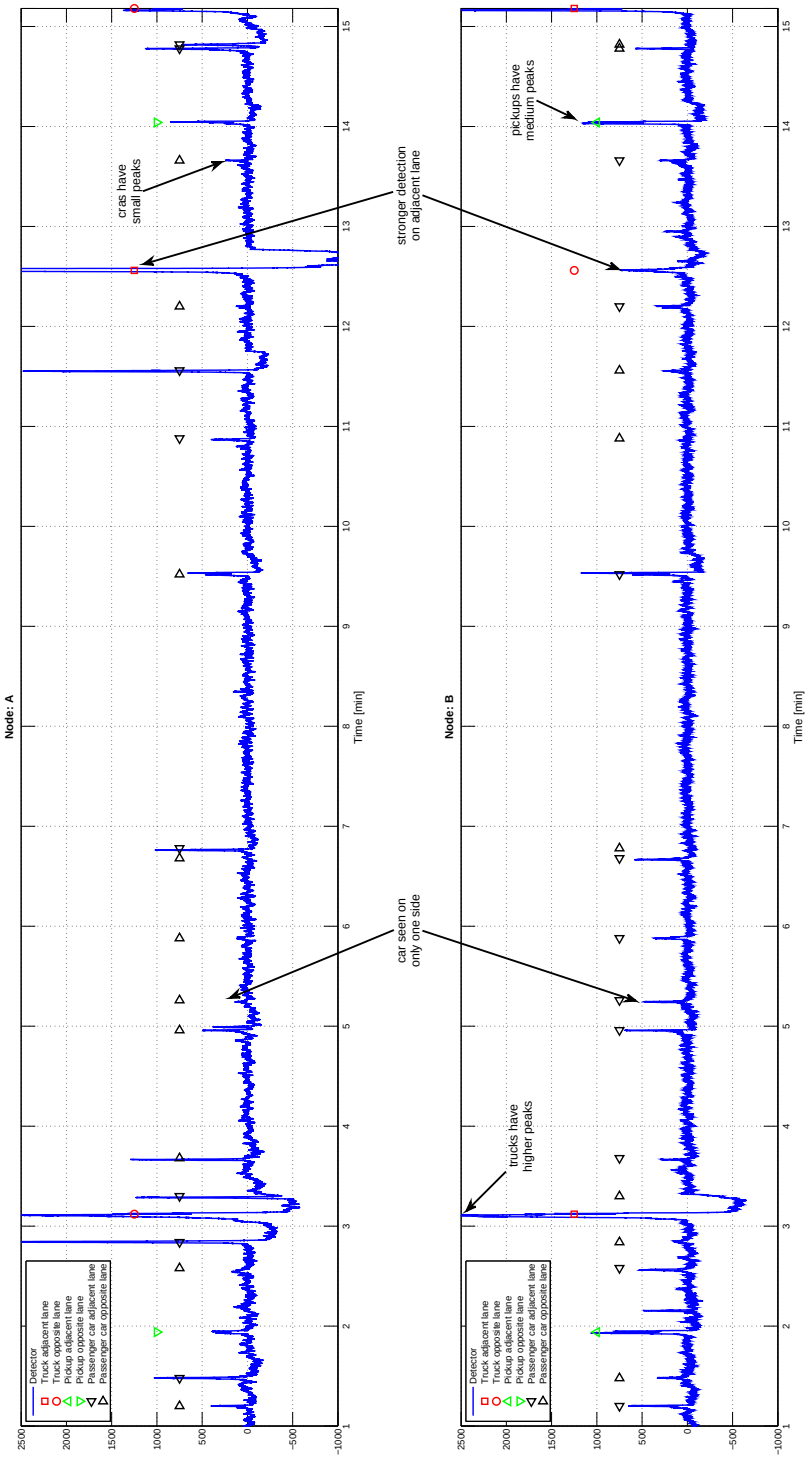


Fig. 9: Illustration of the detection algorithm: trucks can be easier detected than passenger cars. For the latter a cooperative detection algorithm using the sensor data from multiple sensors is required (sensor fusion).

Evaluation: Fig. 9 shows algorithm results from two of our nodes, placed on different road sides. The used algorithms normalizes the measured accelerations and creates compared moving averages of a short and long sliding window, similar to the algorithm used in earth quake early warning. The figure also shows the potential for cooperative vehicle detection: trucks can be detected on both sides of the road, while smaller vehicles are only visible on one side.

6.3 Future Work

The described experiments can only be a first step. In the future, we have to extend experiments to different traffic situations, employ larger amounts of sensors, and combine different types of sensors (sensor-fusion). Similar to earth quake early warning, we expect that the cooperative use of many sensors allows us to increase the quality of our technique. Furthermore, we have to improve our research methodology. A test-road for new vehicle detection methods operated by the *Deutsches Zentrum für Luft und Raumfahrt* (DLR) will allow us to experiment in a more efficient environment and automatically acquired controll data enables us to work with data sets of more statistically relevant sizes.

7 Conclusions

We identified different technologies for smart city applications: wireless sensor networks, geo-information systems, and frameworks for data analysis. While all these technologies exists, it is still a challenge to provide a concise smart city platform that allows developers to use all these technologies together. We successfully used our sensor network HWL to concept proof the applications early earthquake warning and traffic surveillance; we are also able to represent sensor and geo-spatial data within the same infrastructure; and there are a myriad studies on processing large amounts of geo-spatial data. But a large case study that combines all necessary technologies and proofs the practical development of smart city applications is still an open subject, not only for us, but for the research community at large.

References

1. Akyildiz, I.F., Melodia, T., Chowdhury, K.R.: A survey on wireless multimedia sensor networks. Comput. Netw. 51(4), 921–960 (Mar 2007)
2. Akyildiz, I.F., Su, W., Sankarasubramaniam, Y., Cayirci, E.: Wireless sensor networks: a survey. Computer Networks 38(4), 393–422 (2002)
3. Baar, M., Will, H., Blywis, B., Liers, A., Wittenburg, G., Schiller, J.: The ScatterWeb MSB-A2 Platform for Wireless Sensor Networks. Tech. rep., Freie Universität Berlin (2008)
4. Blywis, B., Günes, M., Juraschek, F., Schmidt, P., Kumar, P.: DES-SERT: A framework for structured routing protocol implementation. In: IFIP Wireless Days conference (WD'09). Paris, France (12 2009)

5. Chatzigiannakis, I., Fischer, S., Koninis, C., Mylonas, G., Pfisterer, D.: Wisebed: An open large-scale wireless sensor network testbed. In: Komninos, N. (ed.) SENSAPPEAL. Lecture Notes of the Institute for Computer Sciences, Social Informatics and Telecommunications Engineering, vol. 29, pp. 68–87. Springer (2009)
6. Erdik, M., Fahjan, Y., Ozel, O., Alcik, H., Mert, A., Gul, M.: Istanbul Earthquake Rapid Response and the Early Warning System. Bulletin of Earthquake Engineering 1, 157–163 (2003)
7. Estrin, D., Girod, L., Pottie, G., Srivastava, M.: Instrumenting the world with wireless sensor networks (2001), `citeseer.ist.psu.edu/estrin01instrumenting.html`
8. Fischer, J., Redlich, J.P., Zschau, J., Milkereit, C., Picozzi, M., Fleming, K., Brumbulli, M., Lichtblau, B., Eveslage, I.: A wireless mesh sensing network for early warning. J. Network and Computer Applications 35(2), 538–547 (2012)
9. Fleming, K., Picozzi, M., Milkereit, C., Kühnlenz, F., Lichtblau, B., Fischer, J., Zulfikar, C., Ozel, O., et al.: The Self-organizing Seismic Early Warning Information Network (SOSEWIN). Seismological Research Letters 80(5), 755 (2009)
10. Guenes, M., Juraschek, F., Blywis, B., Mushtaq, Q., Schiller, J.: A testbed for next generation wireless network research (2009), submitted to PiK - Special Issue on Mobile Ad-hoc Networks
11. Günes, M., Blywis, B., Juraschek, F.: Concept and design of the hybrid distributed embedded systems testbed. Tech. Rep. TR-B-08-10, Freie Universität Berlin (August 2008), `ftp://ftp.inf.fu-berlin.de/pub/reports/tr-b-08-10.pdf`
12. Günes, M., Blywis, B., Juraschek, F., Schmidt, P.: Practical issues of implementing a hybrid multi-nic wireless mesh-network. Tech. Rep. TR-B-08-11, Freie Universität Berlin (August 2008), `ftp://ftp.inf.fu-berlin.de/pub/reports/tr-b-08-11.pdf`
13. Hernández-Muñoz, J.M., Vercher, J.B., Muñoz, L., Galache, J.A., Presser, M., Gómez, L.A.H., Pettersson, J.: Smart cities at the forefront of the future internet. In: Domingue, J., Galis, A., et al. (eds.) Future Internet Assembly. Lecture Notes in Computer Science, vol. 6656, pp. 447–462. Springer (2011)
14. Horiuchi, S., Negishi, H., Abe, K., Kamimura, A., Fujinawa, Y.: An Automatic Processing System for Broadcasting Earthquake Alarms. Bulletin of the Seismological Society of America 95(2), 708–718 (2005)
15. Ionescu, C., Böse, M., Wenzel, F., Marmureanu, A., Grigore, A., Marmureanu, G.: An Early Warning System for Deep Vrancea (Romania) Earthquakes. In: Gasparini, P., Manfredi, G., Zschau, J. (eds.) Earthquake Early Warning Systems, pp. 343–349. Springer Berlin Heidelberg (2007)
16. Kohler, E., Morris, R., Chen, B., Jannotti, J., Kaashoek, M.F.: The click modular router. ACM Trans. Comput. Syst. 18(3), 263–297 (2000)
17. Kolovos, D.S., Rose, L.M., Williams, J.R., Matragkas, N.D., Paige, R.F.: A lightweight approach for managing xml documents with mde languages. In: Vallecillo, A., Tolvanen, J.P., Kindler, E., Störrle, H., Kolovos, D.S. (eds.) ECMFA. Lecture Notes in Computer Science, vol. 7349, pp. 118–132. Springer (2012)
18. Lynch, J.P.: A summary review of wireless sensors and sensor networks for structural health monitoring. The Shock and Vibration Digest 38(2), 91–128 (2006)
19. Murty, R., Mainland, G., Rose, I., Chowdhury, A.R., Gosain, A., Bers, J., Welsh, M.: Citysense: An urban-scale wireless sensor network and testbed. In: 2008 IEEE International Conference on Technologies for Homeland Security (May 2008), `http://www.eecs.harvard.edu/~mdw/papers/citysense-ieeehst08.pdf`
20. Portele, C.: OGC Geography Markup Language (GML) 3.3. Tech. rep., Open Geospatial Consortium (OGC) (2 2012)

21. Samadi, P., Mohsenian-Rad, A., Schober, R., Wong, V.W.S., Jatskevich, J.: Optimal Real-Time Pricing Algorithm Based on Utility Maximization for Smart Grid. In: Smart Grid Communications (SmartGridComm), 2010 First IEEE International Conference on. pp. 415–420 (Oct 2010), http://dx.doi.org/10.1109/SMARTGRID.2010.5622077
22. Scheidgen, M.: EMFFrag – Meta-Model-based Model Fragmentation and Persistence Framework. http://code.google.com/p/emf-fragments (2012)
23. Scheidgen, M., Zubow, A., Sombrutzki, R.: Clickwatch – an experimentation framework for communication network test-beds. In: IEEE Wireless Communications and Networking Conference. IEEE, Paris (2012)
24. Scheidgen, M., Zubow, A., Fischer, J., Kolbe, T.H.: Automatic and Transparent Model Fragmentation for Persisting Large Models. In: France, R., Kazmeier, J., Atkinson, C., Breu, R. (eds.) Model Driven Engineering Languages and Systems, 14th International Conference, MODELS 2012, Insbruck, Austria, Sep 30-Oct 5, Proceedings. LNCS, vol. 7590, pp. 102–118. Springer (2012)
25. Scheidgen, M., Zubow, A., Fischer, J., Kolbe, T.H.: Automatic and transparent model fragmentation for persisting large models. In: France, R., Kazmeier, J., Atkinson, C., Breu, R. (eds.) Models 2012. vol. to appear. Springer Verlag Berlin/Heidelberg (2012)
26. Scheidgen, M., Zubow, A., Sombrutzki, R.: ClickWatch – An Experimentation Framework for Communication Network Test-beds. In: IEEE Wireless Communications and Networking Conference, Paris, France, April 1-4, Proceedings. pp. 3296–3301. IEEE (2012)
27. Scheidgen, M., Zubow, A., Sombrutzki, R.: HWL – A High Performance Wireless Research Network. In: 9th International Conference on Networked Sensing System, Antwerp, Belgium, June 11-14, Proceedings. IEEE (2012)
28. Scheidgen, M., Zubow, A., Sombrutzki, R.: Hwl - a high performance wireless sensor research network. International Conference on Networked Sensing Systems (INSS) (2012)
29. Schröder, S., Zilske, M., Liedtke, G., Nagel, K.: A computational framework for a multi-agent simulation of freight transport activities. Annual Meeting Preprint 12-4152, Transportation Research Board, Washington D.C. (2012), https://svn.vsp.tu-berlin.de/repos/public-svn/publications/vspwp/2011/11-19/, also VSPWP 11-19, see www.vsp.tu-berlin.de/publications
30. Stadler, A.: Making interoperability persistent: A 3D geo database based on CityGML. In: Lee, J., Zlatanova, S. (eds.) Proceedings of the 3rd International Workshop on 3D Geo-Information, pp. 175–192. Springer Verlag, Seoul, Korea (2008)
31. Steinberg, D., Budinsky, F., Paternostro, M., Merks, E.: EMF: Eclipse Modeling Framework 2.0. Addison-Wesley Professional, 2nd edn. (2009)
32. Stepper, E.: Connected Data Objects (CDO). http://www.eclipse.org/cdo/ (2012)
33. Vojdani, A.: Smart Integration. Power and Energy Magazine, IEEE 6(6), 71–79 (2008), http://dx.doi.org/10.1109/MPE.2008.929744
34. Wald, D.J., Worden, B.C., Quitoriano, V., Pankow, K.L.: ShakeMap Manual - Technical Manual, Users Guide and Software Guide. U.S. Geological Survey (2006)
35. Wu, Y.M., Teng, T.l.: A Virtual Subnetwork Approach to Earthquake Early Warning. Bulletin of the Seismological Society of America 92(5), 2008–2018 (2002)
36. Zubow, A., Sombrutzki, R., Scheidgen, M.: A low-cost mimo mesh testbed based on 802.11n. IEEE Wireless Communications and Networking Conference (2012)